```c
//FCFS
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

struct process
{
    int pid, at, bt, ct, ta, wt;
};

void swap(struct process *a, struct process *b)
{
    struct process temp = *a;
    *a = *b;
    *b = temp;
}

void sort_at(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
        {
            if (arr[j].at > arr[j + 1].at)
                swap(&arr[j], &arr[j + 1]);
        }
}
void sort_pid(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
        {
            if (arr[j].pid > arr[j + 1].pid)
                swap(&arr[j], &arr[j + 1]);
        }
}
int main()
{
    int pid;
    float avwt = 0, avta = 0;

    do
    {
        printf("Enter the number of process (upto
%d): ", MAX);
        scanf("%d", &pid);
    } while (pid <= 0 || pid > MAX);

    struct process fcfs[pid];

    printf("Enter process details:\n");
    for (int i = 0; i < pid; i++)
    {
        printf("\nProcessID: %d\n", i + 1);
        fcfs[i].pid = i + 1;

        printf("Arrival Time: ");
        scanf("%d", &fcfs[i].at);

        printf("Burst Time: ");
        scanf("%d", &fcfs[i].bt);
    }

    sort_at(fcfs, pid);

    int current_time = 0;

    fcfs[0].ct = 0;

    for (int i = 0; i < pid; i++)
    {
        if (current_time < fcfs[i].at)
            current_time = fcfs[i].at;

        fcfs[i].ct = current_time + fcfs[i].bt;
        fcfs[i].ta = fcfs[i].ct - fcfs[i].at;
        fcfs[i].wt = fcfs[i].ta - fcfs[i].bt;

        avwt += fcfs[i].wt;
        avta += fcfs[i].ta;

        current_time = fcfs[i].ct;
    }

    avta /= pid;
    avwt /= pid;

    sort_pid(fcfs, pid);

    printf("Process table:\n");
    printf("PID\tAT\tBT\tCT\tTA\tWT\n");
    for (int i = 0; i < pid; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n",
fcfs[i].pid, fcfs[i].at, fcfs[i].bt, fcfs[i].ct,
fcfs[i].ta, fcfs[i].wt);

    printf("Average TA: %f\n", avta/pid);
    printf("Average WT: %f\n", avwt/pid);

    return 0;
}
```

```
//SJF
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

struct process
{
    int pid, at, bt, ct, ta, wt;
};

void swap(struct process *a, struct process *b)
{
    struct process temp = *a;
    *a = *b;
    *b = temp;
}

void sort_at(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].at > arr[j + 1].at)
                swap(&arr[j], &arr[j + 1]);
}

void sort_bt(struct process arr[], int n)
{
    int k = 1, min = 0, btime = 0;
    for (int i = 0; i < n; i++)
    {
        btime = btime + arr[i].bt;
        min = arr[k].bt;
        for (int j = k; j < n; j++)
            if (btime >= arr[j].at && arr[j].bt <
min)
                swap(&arr[j], &arr[j - 1]);
        k++;
    }
}

void sort_pid(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].pid > arr[j + 1].pid)
                swap(&arr[j], &arr[j + 1]);
}

int main()
{
    int pid, avta = 0, avwt = 0;

    do
    {
        printf("Enter process upto %d: ", MAX);
        scanf("%d", &pid);
    } while (pid <= 0 || pid > MAX);

    struct process sjf[pid];

    printf("Enter process details: ");
    for (int i = 0; i < pid; i++)
    {
        printf("\nProcess ID: %d", i + 1);
        sjf[i].pid = i + 1;

        printf("\nArrival Time: ");
        scanf("%d", &sjf[i].at);

        printf("Burst Time: ");
        scanf("%d", &sjf[i].bt);
    }

    sort_at(sjf, pid);
    sort_bt(sjf, pid);

    int current_time = 0;
    sjf[0].ct = 0;

    for (int i = 0; i < pid; i++)
    {
        if (current_time < sjf[i].at)
            current_time = sjf[i].at;

        sjf[i].ct = current_time + sjf[i].bt;
        sjf[i].ta = sjf[i].ct - sjf[i].at;
        sjf[i].wt = sjf[i].ta - sjf[i].bt;

        avta += sjf[i].ta;
        avwt += sjf[i].wt;

        current_time = sjf[i].ct;
    }

    avwt = avwt / pid;
    avta = avta / pid;

    sort_pid(sjf, pid);

    printf("Process table:\n");
    printf("PID\tAT\tBT\tCT\tTA\tWT\n");
    for (int i = 0; i < pid; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n",
sjf[i].pid, sjf[i].at, sjf[i].bt, sjf[i].ct,
sjf[i].ta, sjf[i].wt);

    printf("Average TA: %f\n", avta/pid);
    printf("Average WT: %f\n", avwt/pid);

    return 0;
}
```

```c
//Round Robin
#include <stdio.h>
#define MAX 20

struct process
{
    int pid, at, bt, ct, ta, wt, rt;
};

void swap(struct process *a, struct process *b)
{
    struct process temp = *a;
    *a = *b;
    *b = temp;
}

void sort_at(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].at > arr[j + 1].at)
                swap(&arr[j], &arr[j + 1]);
}

void sort_pid(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].pid > arr[j + 1].pid)
                swap(&arr[j], &arr[j + 1]);
}

int main()
{
    int i, j, k, pid, tq, tct, rear, flag = 0;
    float avwt = 0, avta = 0;

    do
    {
        printf("Enter process upto %d:", MAX);
        scanf("%d", &pid);
    } while (pid <= 0 || pid > MAX);

    printf("Enter time quantum: ");
    scanf("%d", &tq);

    struct process rr[pid], temp, q[100];

    printf("Enter process details: ");
    for (int i = 0; i < pid; i++)
    {
        printf("\nProcess ID: %d", i + 1);
        rr[i].pid = i + 1;

        printf("\nArrival Time: ");
        scanf("%d", &rr[i].at);

        printf("Burst Time: ");
        scanf("%d", &rr[i].bt);

        rr[i].rt = rr[i].bt;
    }

    sort_at(rr, pid);

    tct = rr[0].at;
    i = tct + 1;
    j = 0;
    q[0] = rr[0];
    rear = 0;

    while (rear != -1)
    {
        temp = q[0];
        if (rear != 0)
            for (int m = 0; m < rear; m++)
                q[m] = q[m + 1];
        rear--;
        if (temp.rt >= tq)
        {
            tct += tq;
            temp.rt -= tq;
        }
        else
        {
            tct += temp.rt;
            temp.rt = 0;
        }
        if (temp.rt == 0)
        {
            flag = 1;
            temp.ct = tct;
        }
        if (j < pid)
        {
            for (; i <= tct; i++)
                if (rr[j + 1].at == i)
                {
                    j++;
                    rear++;
                    q[rear] = rr[j];
                }
        }
        if (flag == 0)
        {
            rear++;
            q[rear] = temp;
        }
        for (k = 0; k < pid; k++)
            if (temp.pid == rr[k].pid)
                rr[k] = temp;
        flag = 0;
    }

    sort_pid(rr, pid);

    for (i = 0; i < pid; i++)
    {
        rr[i].ta = rr[i].ct - rr[i].at;
        rr[i].wt = rr[i].ta - rr[i].bt;
        avta += rr[i].ta;
        avwt += rr[i].wt;
    }

    printf("Process table:\n");
    printf("PID\tAT\tBT\tCT\tTA\tWT\n");
    for (int i = 0; i < pid; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n",
rr[i].pid, rr[i].at, rr[i].bt, rr[i].ct, rr[i].ta,
rr[i].wt);

    printf("Average TA: %f\n", avta/pid);
    printf("Average WT: %f\n", avwt/pid);

    return 0;
}
```

```
//SRTF
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

struct process
{
    int pid, at, bt, ct, ta, wt, rt;
};

void swap(struct process *a, struct process *b)
{
    struct process temp = *a;
    *a = *b;
    *b = temp;
}

void sort_at(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].at > arr[j + 1].at)
                swap(&arr[j], &arr[j + 1]);
}

void sort_pid(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].pid > arr[j + 1].pid)
                swap(&arr[j], &arr[j + 1]);
}

int main()
{
    int pid;
    float avwt = 0, avta = 0;

    do
    {
        printf("Enter the number of processes (up
to %d): ", MAX);
        scanf("%d", &pid);
    } while (pid <= 0 || pid > MAX);

    struct process srtf[pid];

    printf("Enter process details:\n");
    for (int i = 0; i < pid; i++)
    {
        printf("\nProcessID: %d\n", i + 1);
        srtf[i].pid = i + 1;

        printf("Arrival Time: ");
        scanf("%d", &srtf[i].at);

        printf("Burst Time: ");
        scanf("%d", &srtf[i].bt);

        srtf[i].rt = srtf[i].bt;
    }

    sort_at(srtf, pid);

    int time = 0, completed = 0, short_job;

    while (completed != pid)
    {
        short_job = -1;
        for (int i = 0; i < pid; i++)
            if (srtf[i].at <= time && srtf[i].rt >
0)
                if (short_job == -1 || srtf[i].rt <
srtf[short_job].rt)
                    short_job = i;

        if (short_job != -1)
        {
            srtf[short_job].rt--;
            if (srtf[short_job].rt == 0)
            {
                srtf[short_job].ct = time + 1;
                completed++;
                srtf[short_job].ta =
srtf[short_job].ct - srtf[short_job].at;
                srtf[short_job].wt =
srtf[short_job].ta - srtf[short_job].bt;
                avta += srtf[short_job].ta;
                avwt += srtf[short_job].wt;
            }
        }
        time++;
    }

    sort_pid(srtf, pid);

    printf("\nProcess Table:\n");
    printf("PID\tAT\tBT\tCT\tTA\tWT\n");
    for (int i = 0; i < pid; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n",
srtf[i].pid, srtf[i].at, srtf[i].bt, srtf[i].ct,
srtf[i].ta, srtf[i].wt);

    printf("Average TA: %f\n", avta/pid);
    printf("Average WT: %f\n", avwt/pid);

    return 0;
}
```

```c
//Pre-emptive Priority
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

struct process
{
    int pid, at, bt, prio, ct, ta, wt, rt;
};

void swap(struct process *a, struct process *b)
{
    struct process temp = *a;
    *a = *b;
    *b = temp;
}

void sort_prio(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].prio > arr[j + 1].prio)
                swap(&arr[j], &arr[j + 1]);
}

void sort_pid(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].pid > arr[j + 1].pid)
                swap(&arr[j], &arr[j + 1]);
}

int main()
{
    int pid;
    float avta = 0, avwt = 0;

    do
    {
        printf("Enter process upto %d: ", MAX);
        scanf("%d", &pid);
    } while (pid <= 0 || pid > MAX);

    struct process pprio[pid]; // pprio stands for
preemptive priority

    printf("Enter Process Details: ");
    for (int i = 0; i < pid; i++)
    {
        printf("\nProcess ID: %d", i + 1);
        pprio[i].pid = i + 1;

        printf("\nArrival Time: ");
        scanf("%d", &pprio[i].at);

        printf("Burst Time: ");
        scanf("%d", &pprio[i].bt);

        printf("Priority: ");
        scanf("%d", &pprio[i].prio);

        pprio[i].rt = pprio[i].bt;
    }

    sort_prio(pprio, pid);

    int i = 0, j, completed = 0, max_prio, time =
0;

    while (completed != pid)
    {
        max_prio = 10000;
        int max_rt = -1;
        for (i = 0; i < pid; i++)
            if (pprio[i].at <= time && pprio[i].rt
> 0 && pprio[i].prio < max_prio)
            {
                max_prio = pprio[i].prio;
                max_rt = pprio[i].rt;
                j = i;
            }
        if (max_rt == -1)
        {
            time++;
            continue;
        }
        pprio[j].rt--;
        if (pprio[j].rt == 0)
        {
            pprio[j].ct = time + 1;
            completed++;
            pprio[j].ta = pprio[j].ct -
pprio[j].at;
            pprio[j].wt = pprio[j].ta -
pprio[j].bt;
            avta += pprio[j].ta;
            avwt += pprio[j].wt;
        }
        time++;
    }

    sort_pid(pprio, pid);

    printf("\nProcess Table:\n");
    printf("PID\tAT\tBT\tPrio\tCT\tTA\tWT\n");
    for (int i = 0; i < pid; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
pprio[i].pid, pprio[i].at, pprio[i].bt,
pprio[i].prio, pprio[i].ct, pprio[i].ta,
pprio[i].wt);

    printf("Average TA: %f\n", avta/pid);
    printf("Average WT: %f\n", avwt/pid);

    return 0;
}
```

```c
//Non-Pre-emptive Priority
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

struct process
{
    int pid, at, bt, prio, ct, ta, wt;
};

void swap(struct process *a, struct process *b)
{
    struct process temp = *a;
    *a = *b;
    *b = temp;
}

void sort_prio(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].prio > arr[j + 1].prio)
                swap(&arr[j], &arr[j + 1]);
}

void sort_pid(struct process arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j].pid > arr[j + 1].pid)
                swap(&arr[j], &arr[j + 1]);
}

int main()
{
    int pid, i, j, min_at = INT_MAX;

    do
    {
        printf("Enter process upto %d: ", MAX);
        scanf("%d", &pid);
    } while (pid <= 0 || pid > MAX);

    struct process npprio[pid]; // npprio stands
for non-preemptive priority

    printf("Enter Process Details: ");
    for (int i = 0; i < pid; i++)
    {
        printf("\nProcess ID: %d", i + 1);
        npprio[i].pid = i + 1;

        printf("\nArrival Time: ");
        scanf("%d", &npprio[i].at);

        printf("Burst Time: ");
        scanf("%d", &npprio[i].bt);

        printf("Priority: ");
        scanf("%d", &npprio[i].prio);

        if (min_at > npprio[i].at)
            min_at = npprio[i].at;
    }

    sort_prio(npprio, pid);

    npprio[0].wt = 0;
    npprio[0].ct = npprio[0].bt + npprio[0].at;
    npprio[0].ta = npprio[0].ct - npprio[0].at;
    float avta = npprio[i].ta, avwt = npprio[i].wt;

    for (i = 1; i < pid; i++)
    {
        npprio[i].wt = npprio[i - 1].ct -
npprio[i].at;
        if (npprio[i].wt < 0)
            npprio[i].wt = 0;

        npprio[i].ct = npprio[i].bt + npprio[i -
1].ct;
        npprio[i].ta = npprio[i].ct - npprio[i].at;

        avta += npprio[i].ta;
        avwt += npprio[i].wt;
    }

    sort_pid(npprio, pid);

    printf("\nProcess Table:\n");
    printf("PID\tAT\tBT\tPrio\tCT\tTA\tWT\n");
    for (int i = 0; i < pid; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
npprio[i].pid, npprio[i].at, npprio[i].bt,
npprio[i].prio, npprio[i].ct, npprio[i].ta,
npprio[i].wt);

    printf("Average TA: %f\n", avta/pid);
    printf("Average WT: %f\n", avwt/pid);

    return 0;
}
```